

Mixed Logical and Probabilistic Reasoning for Planning and Explanation Generation in Robotics

Zenon Colaco and Mohan Sridharan

Department of Electrical and Computer Engineering, The University of Auckland, NZ
 zncolaco@gmail.com; m.sridharan@auckland.ac.nz

Abstract. Robots assisting humans in complex domains have to represent knowledge and reason at both the sensorimotor level and the social level. The architecture described in this paper couples the non-monotonic logical reasoning capabilities of a declarative language with probabilistic belief revision, enabling robots to represent and reason with qualitative and quantitative descriptions of knowledge and degrees of belief. Specifically, incomplete domain knowledge, including information that holds in all but a few exceptional situations, is represented as a Answer Set Prolog (ASP) program. The answer set obtained by solving this program is used for inference, planning, and for jointly explaining (a) unexpected action outcomes due to exogenous actions and (b) partial scene descriptions extracted from sensor input. For any given task, each action in the plan contained in the answer set is executed probabilistically. The subset of the domain relevant to the action is identified automatically, and observations extracted from sensor inputs perform incremental Bayesian updates to a belief distribution defined over this domain subset, with highly probable beliefs being committed to the ASP program. The architecture’s capabilities are illustrated in simulation and on a mobile robot in the context of a robot waiter operating in the dining room of a restaurant.

1 Introduction

Robots collaborating with humans in complex domains receive far more raw sensor data than can be processed in real-time. The information extracted from the sensor inputs can be represented probabilistically to quantitatively model the associated uncertainty (“90% certain I saw the book on the shelf”). Robots also receive useful commonsense knowledge that is difficult to represent quantitatively (“books are usually in the library”), and human participants may not have the time and expertise to provide elaborate and accurate feedback. To collaborate with humans, these robots thus need to represent knowledge and reason at both the cognitive level and the sensorimotor level. This objective maps to fundamental research challenges in knowledge representation and reasoning. The architecture described in this paper exploits the complementary strengths of non-monotonic logical reasoning and probabilistic belief revision as a significant step towards addressing these challenges. Specifically, the commonsense logical reasoning capabilities of Answer Set Prolog (ASP), a declarative language, is coupled with probabilistic belief updates, to support the following key features:

- An ASP program represents incomplete domain knowledge, including information that holds in all but a few exceptional situations. The *answer set* obtained by solving the ASP program is used for planning and jointly (a) explaining unexpected

action outcomes by reasoning about exogenous actions; and (b) identifying object occurrences that best explain partial scene descriptions obtained from sensor inputs.

- For any given task, each action in the plan created by inference in the ASP program is executed probabilistically. The relevant subset of the domain (for this action) is identified automatically, and the sensor observations perform incremental Bayesian updates to a belief distribution defined over this subset of the domain, committing highly probability beliefs as statements to the ASP program.

The architecture thus enables robots to represent and reason with qualitative and quantitative descriptions of knowledge and degrees of belief. In this paper, the architecture’s capabilities are demonstrated in simulation and on a mobile robot, in the context of a robot waiter operating in the dining room of a restaurant.

2 Related Work

Knowledge representation, planning and explanation generation are well-researched areas in robotics and artificial intelligence. Logic-based representations and probabilistic graphical models have been used to plan sensing, navigation and interaction for robots and agents. Formulations based on probabilistic representations (by themselves) make it difficult to perform commonsense reasoning, while classical planning algorithms and logic programming tend to require considerable prior knowledge of the domain and the agent’s capabilities, and make it difficult to merge new, unreliable information with an existing knowledge base. For instance, the non-monotonic logical reasoning capabilities of ASP [3] have been used for tasks such as reasoning by simulated robot housekeepers [2] and coordination of robot teams [11]. However, ASP does not support probabilistic analysis of uncertainty, whereas a lot of information extracted from sensors and actuators on robots is represented probabilistically.

Approaches for generating explanations (e.g., through abductive inference or plan diagnosis) use the systems description and observations of system behavior to explain unexpected symptoms [3,9], or use weaker system descriptions and depend on heuristic representation of intuition and past experience [6,8]. Probabilistic and first-order logic-based representations have been combined for better abductive inference [12]. Researchers have also designed architectures for robots that combine deterministic and probabilistic algorithms for task and motion planning [4], combine declarative programming and continuous-time planners for path planning in robot teams [11], or combine a probabilistic extension of ASP with partially observable Markov decision processes (POMDPs) for human-robot dialog [15]. Some principled algorithms that combine logical and probabilistic reasoning include Markov logic network [10], Bayesian logic [7], and probabilistic extensions to ASP [1,5]. However, algorithms based on first-order logic do not provide the desired expressiveness for modeling uncertainty, e.g., it is not always possible to express degrees of belief quantitatively. Algorithms based on logic programming do not support one or more of the desired capabilities such as incremental revision of (probabilistic) information; reasoning as in causal Bayesian networks; and reasoning with large probabilistic components. Towards addressing these challenges, our prior work developed architectures that couple declarative programming and probabilistic graphical models for logical inference, deterministic and probabilistic planning on robots [13,14]. This paper retains the coupling between logical and probabilistic

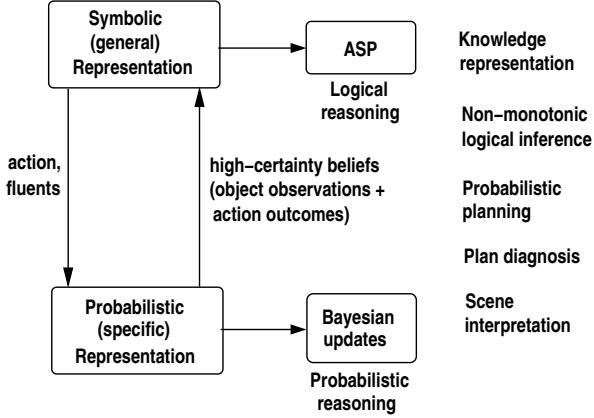


Fig. 1. An overview of the architecture that combines the complementary strengths of declarative programming and probabilistic graphical models for inference, planning, and diagnosis.

reasoning but significantly expands the capabilities of these architectures to support: (1) explanation of unexpected action outcomes and the partial descriptions extracted from sensor inputs; and (2) representation and reasoning at a higher resolution using ASP, making the probabilistic reasoning more computationally efficient.

3 Proposed Architecture

Figure 1 is an overview of the mixed architecture. The symbolic representation is translated to an Answer Set Prolog (ASP) program used for non-monotonic logical inference and planning a sequence of actions for any given task. For each action, the relevant subset of the domain is defined automatically. Sensor observations perform incremental Bayesian updates to a probability distribution over this domain subset, committing high probability beliefs (action outcomes, observation of object attributes) as statements to the ASP program. Observed unexpected action outcomes are explained by reasoning about exogenous actions, and objects are identified to best explain the partial descriptions extracted from visual cues. ASP-based representation and reasoning is performed at a resolution that provides high reliability while also simplifying the (coupled) probabilistic reasoning and tailoring it to specific actions.

The syntax, semantics and representation of the transition diagrams of the architecture's domain representation are described in an *action language* AL [3]. AL has a sorted signature containing three *sorts*: *statics*, *fluents* and *actions*. Statics are domain properties whose truth values cannot be changed by actions, fluents are properties whose values are changed by actions, and actions are elementary actions that can be executed in parallel. AL allows three types of statements:

$a \text{ causes } l_b \text{ if } p_0, \dots, p_m$	(Causal law)
$l \text{ if } p_0, \dots, p_m$	(State constraint)
$\text{impossible } a_0, \dots, a_k \text{ if } p_0, \dots, p_m$	(Executability condition)

where a is an action, l is a literal, l_b is a basic fluent (also called inertial fluent) literal, and p_0, \dots, p_m are domain literals (any domain property or its negation). A collection of statements of AL forms a system description.

As an illustrative example used in this paper, consider a robot waiter that greets and seats people at tables in a restaurant, and delivers orders. The sorts of the domain are arranged hierarchically, e.g., *location* and *thing* are subsorts of *entity*; *animate* and *inanimate* are subsorts of *thing*; *person* and *robot* are subsorts of *animate*; *object* is a subsort of *inanimate*; and *room*, *area*, *door*, and *floor* are subsorts of *location*. We include specific rooms, e.g., *kitchen* and *dining*, and consider objects of sorts such as *table*, *chair* and *plate*, to be characterized by attributes *size*, *color*, *shape*, and *location*. The sort *step* is included for temporal reasoning.

ASP Domain Representation: The ASP program is based on a domain representation that includes a system description \mathcal{D}_H and a history with defaults \mathcal{H} . \mathcal{D}_H has a sorted signature $\Sigma_H = \langle \mathcal{O}, \mathcal{F}, \mathcal{P} \rangle$ that defines the names of objects, functions, and predicates available for use, and axioms that describe a transition diagram τ_H . Fluents are defined in terms of the sorts of their arguments, e.g., *has_location(thing, location)* *in_hand(robot, object)*, *is_open(door)*, and *can_move(robot, location)*. The first three are *basic fluents* that obey the laws of inertia and can be changed directly by actions; the last one is a *defined fluent* that is not subject to inertia and cannot be changed directly by an action. Statics such as *connected(location, location)* and *belongs(location, location)* specify connections between locations, relation *holds(fluent, step)* implies a specific fluent holds at a specific timestep, and *occurs(action, step)* (hypothesizes) that a specific action occurs at a specific timestep. We include *actions*, e.g., *move(robot, location)*, *seat_person(robot, person, table)*, *search_person(robot, area)*, *pickup(robot, object)*, *putdown(robot, object)*, and define domain dynamics using causal laws such as:

$$\begin{aligned} move(R, L) &\text{ causes } has_location(R, L) \\ pickup(R, O) &\text{ causes } in_hand(R, O) \\ open(R, D) &\text{ causes } is_open(D) \\ seat_person(R, P, T) &\text{ causes } has_location(P, L) \text{ if } has_location(T, L) \end{aligned} \tag{1}$$

state constraints such as:

$$\begin{aligned} has_location(O, L) &\text{ if } has_location(R, L), in_hand(R, O) \\ \neg has_location(Th, L_2) &\text{ if } has_location(Th, L_1), L_1 \neq L_2 \\ has_location(Th, L_2) &\text{ if } has_location(Th, L_1), belongs(L_1, L_2) \\ can_move(R, L_2) &\text{ if } has_location(R, L_1), connected(L_1, L_2) \end{aligned} \tag{2}$$

and executability conditions such as:

$$\begin{aligned} \mathbf{impossible} \quad move(R, L) &\text{ if } has_location(R, L) \\ \mathbf{impossible} \quad pickup(R, O) &\text{ if } has_location(R, L_1), has_location(O, L_2), L_1 \neq L_2 \\ \mathbf{impossible} \quad open(R, D) &\text{ if } is_open(D) \end{aligned} \tag{3}$$

Since robots frequently receive *default* domain knowledge that is true in all but a few exceptional situations, the domain history \mathcal{H} , in addition to *hpd(action, step)* and *obs(fluent, boolean, step)*, the occurrence of specific actions and the observation of specific fluents at specific time steps, contains prioritized defaults describing the values of fluents in their initial states. For instance, it may be initially believed that dishes to be delivered are typically on a table between the kitchen and the dining room—if they are

not there, they are still in the kitchen. Existing definitions of entailment and consistency are used to reason with such histories, and any observed exceptions [13].

The domain representation is translated into a program $\Pi(\mathcal{D}_H, \mathcal{H})$ in CR-Prolog that incorporates consistency restoring rules in ASP [3]. Π includes the causal laws of \mathcal{D}_H , inertia axioms, closed world assumption for actions and defined fluents, reality checks, and records of observations, actions and defaults from \mathcal{H} . Every default is turned into an ASP rule and a consistency-restoring (CR) rule that allows us to assume the default's conclusion is false to restore Π 's consistency. ASP is based on stable model semantics, introduces concepts such as default negation and epistemic disjunction, and represents recursive definitions, defaults, causal relations, and language constructs that are difficult to express in classical logic formalisms. The ground literals in an *answer set* obtained by solving Π represent beliefs of an agent associated with Π —statements that hold in all such answer sets are program consequences. Inference and planning can be reduced to computing answer sets of program Π by adding a goal, a constraint stating that the goal must be achieved, and a rule generating possible future actions.

Our architecture supports reasoning about exogenous actions to explain the unexpected (observed) outcomes of actions. For instance, to reason about a door between the kitchen and the dining room being locked by a human, and to reason about a person moving away from a known location, we introduce exogenous actions *locked(door)* and *moved_from(person,location)* respectively, and suitably add (or revise) axioms:

$$\begin{aligned} is_open(D) &\leftarrow open(R,D), \neg ab(D) \\ ab(D) &\leftarrow locked(D) \\ \neg has_location(P,L) &\leftarrow moved_from(P,L), has_location(P,L) \end{aligned} \tag{4}$$

where a door is considered *abnormal* if it has been locked, say by a human. We also introduce an *explanation generation* rule and a new relation *expl*:

$$\begin{aligned} occurs(A,I) | \neg occurs(A,I) &\leftarrow exogenous_action(A), I < n \\ expl(A,I) &\leftarrow action(exogenous,A), occurs(A,I), not hpd(A,I) \end{aligned} \tag{5}$$

where *expl* holds if an exogenous action is hypothesized but there is no matching record in the history. We also include *awareness* axioms and *reality check* axioms:

$$\begin{aligned} holds(F,0) \text{ or } \neg holds(F,0) &\leftarrow fluent(basic,F) \quad \% \text{ awareness axiom} \\ occurs(A,I) &\leftarrow hpd(A,I) \\ \leftarrow obs(fluent,true,I), \neg holds(fluent,I) &\quad \% \text{ reality check} \\ \leftarrow obs(fluent,false,I), holds(fluent,I) &\quad \% \text{ reality check} \end{aligned} \tag{6}$$

The reality check axioms cause a contradiction when observations do not match expectations, and the explanation for such unexpected symptoms can be extracted from the answer set of the corresponding program [3]. The new knowledge is included and used to generate the subsequent plans. This approach provides *all* explanations of an unexpected symptom—using a CR rule instead of the explanation generation rule (in Statement 5) provides the minimal explanation (see below).

A robot processing sensor inputs (e.g., camera images) is typically able to extract partial descriptions of scene objects. The proposed architecture also identifies object

occurrences that best explain these partial descriptions. In our illustrative example, we introduce static relations to establish object class membership¹ and introduce relations to capture ideal (and default) definitions of object attributes, e.g., for a table:

$$\begin{aligned} \text{has_color}(O, \text{white}) &\leftarrow \text{member}(O, \text{table}) \\ \text{has_size}(O, \text{medium}) &\leftarrow \text{member}(O, \text{table}) \\ \text{has_wheels}(4) &\leftarrow \text{member}(O, \text{table}), \neg \text{has_location}(O, \text{kitchen}) \end{aligned} \quad (7)$$

where tables usually have wheels except in the kitchen. Similarly, for a chair:

$$\begin{aligned} \text{has_color}(O, \text{white}) &\leftarrow \text{member}(O, \text{chair}) \\ \text{has_size}(O, \text{medium}) &\leftarrow \text{member}(O, \text{chair}) \\ \neg \text{has_wheels}(O) &\leftarrow \text{member}(O, \text{chair}) \end{aligned} \quad (8)$$

Other objects and object attributes are encoded similarly. As before, a reality check axiom causes an inconsistency when an object does not have a class label due to incomplete information, and a CR rule restores consistency by assigning class labels:

$$\begin{aligned} &\leftarrow \text{object}(O), \text{not } \text{class_known}(O) \quad \% \text{ reality check} \\ \text{is_a}(O, C) &\stackrel{+}{\leftarrow} \text{object}(O) \quad \% \text{ CR rule} \end{aligned} \quad (9)$$

This assignment of a class label to an object is based on the smallest number of rules that need to be relaxed to support the assignment. This information is also added to the ASP program and be used for subsequent reasoning. However, both planning and object recognition are based on processing sensor inputs and moving to specific locations—these tasks are accomplished using probabilistic algorithms, as described below.

Probabilistic Domain Representation: Our previous work created the ASP-based representation at a coarser resolution (e.g., rooms and places), and *refined* it by adding suitable actions, fluents and sorts (e.g., cells in rooms) to define a transition diagram and create its probabilistic version that was modeled as a POMDP [13]. The proposed architecture models the domain at a higher resolution using ASP. For any given task, the ASP-based plan consists of primitive actions that can be executed by the robot. Each such action is executed probabilistically, with the robot maintaining a probability (belief) distribution over the relevant subset of the domain that is identified automatically based on the set of related fluents, e.g., for moving between two tables, the robot only needs to reason about its own location in a subset of areas. The belief distribution is revised incrementally by sensor observations using Bayesian updates. For instance, to update the belief about the location of a specific dish in the dining room:

$$p(E_i|O_i) = \frac{p(O_i|E_i)p(E_i)}{(p(O_i|E_i)p(E_i) + p(O_i|\neg E_i)p(\neg E_i))} \quad (10)$$

where O_i is the event the dish was observed in area i , E_i is the event the dish exists in area i , making $p(O_i|E_i)$ and $p(E_i|O_i)$ are the observation likelihood and the posterior

¹ Relation $\text{member}(object, class)$ is applied recursively in a class hierarchy, $\text{is_a}(object, class)$ denotes an instance of a specific class, and $\text{class_known}(object)$ holds for any object whose class label is known.

probability of existence of the dish in each area in the dining room. The initial knowledge (i.e., prior: $p(E_i)$) is based on domain knowledge or statistics collected in an initial training phase (see Section 4). A Bayesian state estimation approach is used by the robot to estimate its own position (e.g., *particle filters*), navigate, and to process sensor inputs to extract information about objects being observed.

In summary, for any given task, ASP planning provides a plan with deterministic effects. The first action in the plan and relevant information (from ASP inference) identify the subset of the domain to be represented probabilistically, and set the initial (probabilistic) belief distribution. The action is executed probabilistically, updating the belief distribution until a high belief indicates action completion or a time limit is exceeded, and adding relevant statements to the ASP program. Any unexpected action outcomes and partial scene descriptions are explained by reasoning about exogenous actions and possible objects. Once these explanations restore consistency, either the next action (in the plan) is selected for execution or a new plan is created. In what follows, we refer to the proposed architecture as the “mixed architecture”, and compare it with two algorithms: (1) ASP-based reasoning for completing the assigned tasks; and (2) a (greedy) probabilistic approach that maintains a probabilistic belief distribution and heuristically selects actions (and makes decisions) based on the most likely state.

The mixed architecture raises some subtle issues. First, committing probabilistic beliefs above a specific threshold (e.g., 0.85) as fully certain statements to the ASP program may introduce errors, but the non-monotonic reasoning capability of ASP helps the robot recover. Second, with previous work that reasoned at a coarser resolution with ASP and used POMDPs for probabilistic planning (a) computing POMDP policies for each ASP action is computationally expensive; and (b) there may be improper reuse of information if the probabilistic belief distribution is not reset between trials [14]. Third, the mixed architecture presents an interesting trade-off between the resolution of symbolic representation and probabilistic representation. Moving most of the reasoning to a symbolic representation can reduce accuracy and also be computationally expensive—the mixed architecture is a good trade-off of accuracy and efficiency.

4 Experimental Setup and Results

Experiments were conducted in simulation and on a mobile robot in scenarios that mimic a robot waiter in a restaurant. The robot’s tasks include finding, greeting and seating people at tables, and delivering orders appropriately.

4.1 Experimental Setup and Hypotheses

The experimental trials used existing implementations of relevant control and sensor input processing algorithms. In an initial training phase, the robot collected statistics of executing these algorithms to compute the motion error models and observation likelihood models. These models were also used to make the simulation trials more realistic.

The experimental trials considered three hypotheses, evaluating that the mixed architecture: (H1) generates plans for different tasks, and explains unexpected outcomes and partial descriptions extracted from sensor inputs; (H2) significantly improves the task completion accuracy and provides similar task completion time in comparison with

just ASP-based reasoning; and (H3) significantly improves task completion time and provides similar accuracy in comparison with the purely probabilistic approach. We provide both qualitative and quantitative results in simulation and on a mobile robot.

4.2 Experimental Results

The following execution traces demonstrate the planning and diagnosis capabilities.

Example 1 [Explain unexpected action outcome]

The task is to return dish ds_1 to the kitchen from the dining room, e.g., from $table1$ to area a_3 in Figure 2(a). Unknown to the robot, the door d_2 has been locked.

- The robot is in the dining room with the dish in hand:

$$\begin{aligned} & holds(has_location(robot, dining), 0), hold(has_location(robot, area1), 0), \\ & holds(in_hand(robot, ds_1), 0) \end{aligned}$$

The initial plan obtained by computing the answer set is:

$$\begin{aligned} & occurs(move(robot, a_2), 1), occurs(move(robot, d_2), 2), \\ & occurs(open(robot, d_2), 3), occurs(move(robot, a_3), 4), \\ & occurs(putdown(robot, ds_1), 5). \end{aligned}$$

- Each step in this plan, starting with the first one, is executed probabilistically.
- Unfortunately, the robot's attempt to open door d_2 does not produce the expected observation—instead, $obs(is_open(d_2), false, 3)$ is added to the history.
- Diagnosis provides the explanation $expl(locked(d_2), 3)$, which invokes Statement 4 to restore consistency.
- The robot seeks human help to unlock the door, before creating a new plan to successfully return the dish to the kitchen.

Example 2 [Explain partial scene description]

The robot delivering a dish sees a medium-sized white object from a distance, but is unable to assign a class label in the absence of any further information.

- Then initial knowledge consists of:
 $has_size(ob_1, medium), has_color(ob_1, white)$
- Two possible interpretations are generated by Statements 7,8 as an explanation:
 $is_a(ob_1, table)$ or $is_a(ob_1, chair)$.
- As the robot gets closer, it observes: $has_wheel(ob_1, 4), has_location(ob_1, dining)$, i.e., the object has wheels and is in the dining room. Statements 7,8 provide the interpretation: $is_a(ob_1, table)$.

Simulation Experiments: The robot in the simulator had to find and move (seat) dishes (people) to specific locations. Table 1 summarizes the results—each entry is the average of 500 trials. The task, initial position of the robot, and position of objects, were different between the trials, and paired trials were used to establish statistical significance. In each paired trial, for each approach being compared, the initial location of the robot and the location of domain objects are the same. The statistically significant results show that the mixed architecture (a) significantly improves the accuracy and provides similar task completion time in comparison with ASP-based reasoning; and (b) significantly improves task completion time and provides similar accuracy in comparison with the

Table 1. Task completion accuracy and time using only ASP, and using a probabilistic approach, expressed as a factor of the values provided by the mixed architecture. The proposed mixed architecture *significantly* reduces the task completion time while improving the accuracy.

Algorithms	Evaluation metrics	
	Accuracy	Time
ASP only	0.82	1.06 ± 0.6
Probabilistic	0.99	3.32 ± 3.0
Mixed approach	1	1

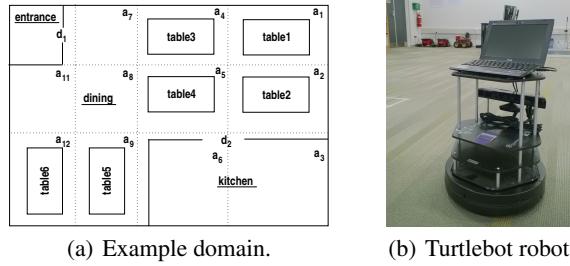


Fig. 2. (a) Example map of illustrative domain used for experimental evaluation, with rooms, doors, and tables (people and robot not shown); and (b) the Turtlebot mobile robot platform.

probabilistic approach. Furthermore, the planning and execution time are significantly reduced in comparison with approaches that combined ASP with probabilistic graphical models [13], while providing comparable accuracy—analysis in other domains may help automate the choice of resolution for symbolic and probabilistic representations for a given task and domain.

Robot Experiments: Trials were conducted on a *Turtlebot* (Figure 2(b)) equipped with a Kinect (RGB-D) sensor, range sensors, and an on-board processor running Ubuntu Linux. Our architecture and algorithms were implemented using the Robot Operating System (ROS). Trials included instances of the domain introduced in Section 3, each with one or more tables, people and other objects, e.g., Figure 2(a). The robot was equipped with probabilistic algorithms to determine the attribute values of objects (e.g., color and shape) from camera images, revise the map of the domain, and determine its location in the map. The robot was able to use the mixed architecture to successfully complete the assigned tasks in all such scenarios, with results of paired trials being similar to those obtained in simulation. A video of an experimental trial showing planning and diagnosis can be viewed online: <https://vimeo.com/130279856>

5 Conclusions

This paper described an architecture that mixes the complementary strengths of declarative programming and probabilistic belief updates. Plans created using ASP-based non-monotonic logical reasoning are implemented probabilistically, with high probability observations and action outcomes adding statements to the ASP program. The architecture enables a robot to explain unexpected action outcomes by reasoning about exogenous actions, and to identify objects that best explain partial scene descriptions. These capabilities have been demonstrated through experimental trials in simulation and on

a mobile robot in scenarios that mimic a robot waiter in a restaurant’s dining room. Future work will further investigate the tight coupling and transfer of control between the logical and probabilistic representations, with the long-term objective of enabling robots to collaborate with humans in complex application domains.

Acknowledgments

The authors thank Michael Gelfond and Rashmica Gupta for discussions that contributed to the development of the architecture described in this paper. This work was supported in part by the US Office of Naval Research Science of Autonomy award N00014-13-1-0766. All opinions and conclusions described in this paper are those of the authors.

References

1. Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming*, 9(1):57–144, January 2009.
2. Esra Erdem, Erdi Aker, and Volkan Patoglu. Answer Set Programming for Collaborative Housekeeping Robotics: Representation, Reasoning, and Execution. *Intelligent Service Robotics*, 5(4):275–291, 2012.
3. Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning and the Design of Intelligent Agents*. Cambridge University Press, 2014.
4. Leslie Kaelbling and Tomas Lozano-Perez. Integrated Task and Motion Planning in Belief Space. *International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.
5. Joohyung Lee and Yi Wang. A Probabilistic Extension of the Stable Model Semantics. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 2015.
6. Ben Meadows, Pat Langley, and Miranda Emery. Seeing Beyond Shadows: Incremental Abductive Reasoning for Plan Understanding. In *AAAI Workshop on Plan, Activity and Intent Recognition*, Bellevue, USA, March 25-27, 2013.
7. Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic Models with Unknown Objects. In *Statistical Relational Learning*. MIT Press, 2006.
8. Hwee Tou Ng and Raymond J. Mooney. Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. In *Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 499–508. Morgan Kaufmann, 1992.
9. Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–95, 1987.
10. Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, February 2006.
11. Zeynep Saribatur, Esra Erdem, and Volkan Patoglu. Cognitive Factories with Multiple Teams of Heterogeneous Robots: Hybrid Reasoning for Optimal Feasible Global Plans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.
12. Sindhu Raghavan and Raymond J. Mooney. Abductive Plan Recognition by Extending Bayesian Logic Programs. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 629–644, September 2011.
13. Shiqi Zhang, Mohan Sridharan, Michael Gelfond, and Jeremy Wyatt. Towards An Architecture for Knowledge Representation and Reasoning in Robotics. In *International Conference on Social Robotics (ICSR)*, pages 400–410, Sydney, Australia, October 27-29, 2014.

14. Shiqi Zhang, Mohan Sridharan, and Jeremy Wyatt. Mixed Logical Inference and Probabilistic Planning for Robots in Uncertain Worlds. *IEEE Transactions on Robotics*, 31(3):699–713, 2015.
15. Shiqi Zhang and Peter Stone. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *AAAI Conference on Artificial Intelligence*, pages 1394–1400, Austin, USA, 2015.